RT-119: Systemic Assurance

Technical Report SERC-2015-TR-019-1

31 July 2015

Principal Investigator:
Dr. William Scherlis, Carnegie Mellon University

Research Team:
Dr. Jonathan Aldrich, Carnegie Mellon University
Dr. Travis D. Breaux, Carnegie Mellon University
Dr. David Garlan, Carnegie Mellon University
Dr. Christian Kästner, Carnegie Mellon University
Dr. Claire Le Goues, Carnegie Mellon University
Dr. Bradley Schmerl, Carnegie Mellon University
Dr. Joshua Sunshine, Carnegie Mellon University

Period of Performance: July 14, 2014 to July 31, 2015

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
31 JUL 2015	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Systemic Assurance	**HQ0034-13-D-0004**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Dr. William Scherlis Dr. Jonathan Aldrich Dr. Travis D. Breaux Dr. David Garlan Dr. Christian KÃ¤stner Dr. Claire Le Goues Dr. Bradley Schmerl Dr. Joshua Sunshine	**RT 119**
	5e. TASK NUMBER **TO 019**
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Carnegie Mellon University	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
Deputy Assistant Secretary for Defense for Systems Engineering, 4800 Mark Center Drive, 17C08, Alexandria, VA	**DASD (SE)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

Assurance, SE Readiness, EM Transformation, The original document contains color images.

14. ABSTRACT

The Systemic Assurance project is to improve the level of assurance and the efficacy and efficiency through which it is achieved. The project is advancing four technical themes related to improving assurance capability: enable composition of assurance judgments for components, use of semantics-based techniques, and integrate methods for requirements of heterogeneous systems. Develop baseline and intervention models for a selection of current standards and practices with DoD stakeholders. This baselining effort is essential to support a measurement-based approach to documenting the impact of the proposed new interventions: Integrating improved capability for traceability and evidence management though integrated development environments with minimal team practice disruption. Implement experiments using assurance-related reasoning for rapid recertification particularly those with externally developed components. Develop a framework for assessment of architecture-derived quality attributes. Develop requirements and management approaches for assurance objectives. Augment existing efforts to address particular quality criteria particularly those outside conventional testing techniques. Identify and advance areas in support of increasing automation with immediate rewards. A key area of emphasis is to continue refinement of the meta-criteria and initiate validation with subject-matter experts. These will lead to a concept document describing potential new-generation assurance practices and incentives.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a REPORT **unclassified**	b ABSTRACT **unclassified**	c THIS PAGE **unclassified**	**UU**	**23**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004.

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor ASD(R&E).

ABSTRACT

Systems cannot be deployed until customer organizations judge them fit for use in the mission environment. These assurance judgments must be based on evidence that a system manifests the necessary functionality and does so at a level of quality and security appropriate to the operating environment. Achieving this goal has two benefits. The first is direct: Cost-effective and rapid recertification is essential to support the development of systems that must adapt to changes in both the mission environment and the infrastructure environment. The second benefit is indirect: The prerequisites for progress towards this goal are the same as those that will, in the more general case, improve the level of assurance and the efficacy and efficiency of the processes through which it is achieved.

Assurance is a human judgment of fitness for use. This judgment is thus contextualized by the mission definition, the features of the normative operating environment, the threat landscape, and characteristics of the evolving infrastructure environment. The judgment must be based on evidence, and there are many different kinds of evidence that can be produced and managed in a large-scale systems engineering activity. The judgment must also address diverse quality attributes and ilities.

The project takes a multi-faceted approach, focusing on combines technical analysis of system artifacts and requirements with architecture techniques to promote assurance and resiliency. An important goal is to apply these techniques not just in anomaly detection, but also to support stronger possibilities for positive assurance, guaranteeing an absence of defects of certain specific kinds, for greater scalability to large systems, building on technical approaches to composability, and for more rapid execution, building on design experience. These have been successfully applied in such areas as the High Level Architecture analysis for networked DoD models and simulations, cyber-physical robotic systems, and extremely large commercial Java programs.

An important goal is to develop incrementally compostable combinations of models, practices, and tools for obtaining the most cost- and schedule-effective combinations for the assurance of necessary system properties. One analogy to exploit is with successful techniques in other domains, such as building codes. Building codes provide engineering guidance and constraint. But they are also continuously evolving, and successfully doing so under the influence of diverse and conflicting stakeholder interests. Another analogy to exploit is the idea of chains of evidence -- semantic dependency modeling -- to support ongoing re-evaluation for rapidly evolving systems both in development and in sustainment/modernization. A third analogy is with the use of metaphors in language -- how, in the engineering and evaluation of systems, do we choose to express the key design concepts and engineering abstractions? And how do these choices influence the kinds of models and analytics and the extent to which assurance judgments can be reached and at what scale of system and complexity? The abstractions come in many forms, including for example language extensions for assurance assertions or context metadata.

TABLE OF CONTENTS

FIGURES AND TABLES

EXECUTIVE SUMMARY

The Systemic Assurance project is advancing four technical themes related to improving assurance capability:

- Evidence and traceability. Facilitate early validation by accumulating assurance-related evidence and creating traceability structures during development.
- Requirements, architecture, composition, variabilities. Address assurance goals in the earliest phases of development. Enable composition of assurance judgments for components into overall judgments for systems.
- Direct analysis. Use semantics-based techniques to enhance confidence and scalability, focusing on challenges significant for modern systems, such as: framework protocol compliance, highly versioned systems, automatic defect repair, safe concurrency, and other areas.
- Combined methods. Integrate multiple methods to evaluate quality attribute requirements for heterogeneous systems, combining informal and formal, static and dynamic, and development and operational monitoring.

The project builds on a baseline examination of a sample of existing practices for acceptance evaluation. DO-178C, for example, which is used by the FAA and other authorities, potentially supports traceability, model-based techniques, formal methods to complement testing, and object oriented technology. The NIAP Common Criteria, another example, evaluates security attributes of designs against intended mission security needs. (See first subtask).

The team addresses these themes through seven subprojects:

Practice Baselining

Develop baseline and intervention models for a selection of current standards and practices (identified in collaboration with DoD stakeholders), refining technical understanding of gaps and limitations. This baselining effort is essential to support a measurement-based approach to documenting the impact of the proposed new technologies and process interventions. This includes identifying the key criteria and dimensions of measurement.

Evidence Building

Undertake engineering design effort focused on integrating improved capability for traceability and other features required to support explicit modeling and management of chains of evidence. A key focus is to demonstrate that it is possible to enhance existing tools and environments, including both integrated development environments (IDEs) and team tools, with relatively little disruption to established team practices and metrics.

Recertification Practice

Design and implement experiments to address the challenge of rapid recertification. These include capturing evidence and assurance-related reasoning (assurance cases, models, analyses, configuration management, etc.). This area of rapid recertification is critical to iterative, incremental, and staged development practices. It is also critical to systems with supply chains that include externally developed

components and infrastructure such as commercial and open-sourced databases, operating systems, frameworks, and libraries. (Almost all larger-scale software-reliant systems have this characteristic.)

Architecture Assessment

Develop a framework for assessment of architecture-derived quality attributes, focusing on architectural modeling and the relationship of architectural and compositional models with quality outcomes. This is essential in order to ensure that key decisions made at early lifecycle phases will have intended quality outcomes.

Requirements and Quality Validation

Develop requirements elicitation and management approaches that better address quality and policy objectives. Requirements elicitation and management is one of the earliest areas of focus in an engineering process, and decisions at this point can have tremendous leverage on quality outcomes. This work is directed at providing more immediate assessments of the potential outcomes of early requirements-related decisions. By improving models, it becomes possible to better manage the linkage of requirements and architectural decisions.

Technical Quality Attributes

Augment and collaborate with diverse existing efforts focused on technical means to address particular quality criteria. Many of these quality criteria are emerging as significant challenges because they tend to defy conventional testing and inspection techniques. These include, for example, a number of attributes related to safe concurrency, compliance with application program interface rules-of-the-road, cyber-physical architectural compliance, state and access management for shared objects, taint and flow and other security-related attributes, and others.

Tools, Automation, and Usability

Identify and advance areas in support of increasing automation, in order to reduce workload of developers and evaluators and to advance existing workload forward in the process, with immediate rewards. The purpose of this is to frame an ultimately more quantitative business case for adoption based on increased return on investment for assurance-related effort and reduced uncertainty (lesser variance in estimate "cones"). This is supportive of the longer-term goal of a "positive benefit" model for the adoption of assurance-related practices. It also supports a stakeholder-engaged process model analogous to building codes.

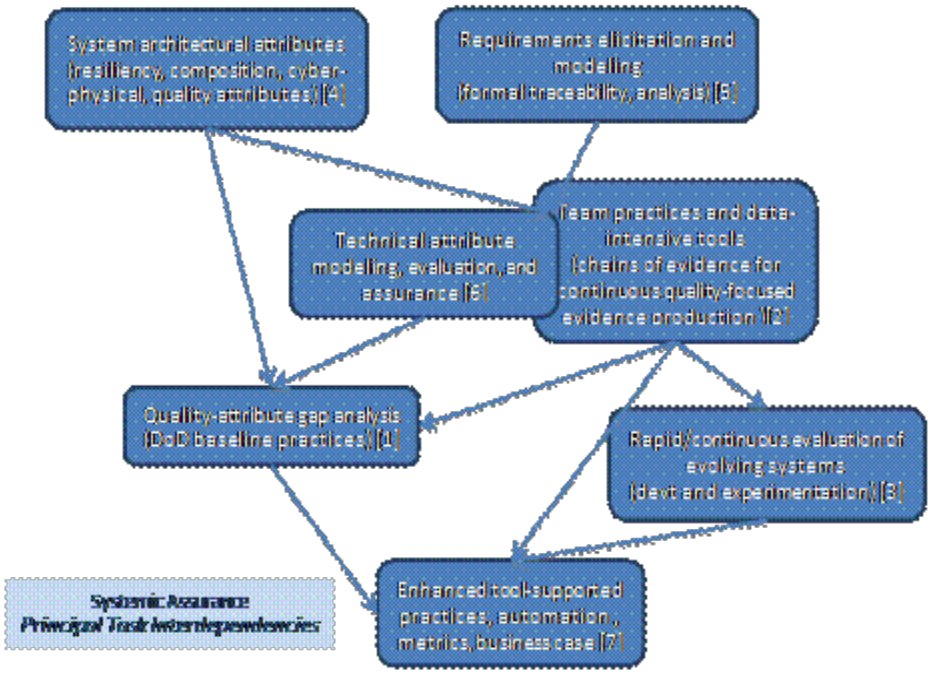

Figure 1: Task Interdependencies

The extent of the assurance challenge is growing rapidly:

- More systems involve complex integrations of hardware, software, and human operators. Reliability is influenced by how software responds to hardware faults and human errors.
- Systems should degrade gracefully when under attack rather than fail outright.
- The operating environment involves interlinking with other systems, working in coalitions, relying on civil infrastructure, and use of personal devices.
- Modern systems, such as mobile and big-data, rely on extensive libraries and frameworks with rich supply chains. These architectures expose attack surfaces within systems.
- Systems are more autonomous, with some rules of engagement embodied in the system and evaluators taking more responsibility for assuring rule compliance.

Prospects for systemic assurance are improving, despite rapid changes in the mission environment, in the technology infrastructure, and in the advancement of new technical ideas in university laboratories (and, to our awareness, a small set of industry laboratories). In this section we summarize some of the challenges faced in attempting to achieve confident assurance judgments.

Characterizing these challenges is an important feature of the project activity, as is the baselining of assurance-evaluation standards. Both of these activities inform the development and weighting of the meta-criteria, according to which we evaluate the benefits, costs, and risks of future frameworks for assurance-evaluation standards.

The challenges are in two categories: relating to the evolution of needs for modern mission systems, and relating to the baseline of current practices in support of making assurance judgments. The first category relates to the characteristics of the systems, and the identified challenges are significant even when best practices are applied.

Challenges of Modern Mission Systems

We note five categories of challenges in modern mission systems:

MODERN SYSTEMS INVOLVE INTERTWINED INTEGRATIONS OF HARDWARE, SOFTWARE, AND HUMAN OPERATORS

Certainly, for the past several decades, functionality has been incrementally migrating from both hardware and human operators into software. This can dramatically reduce variable costs for systems, though it can increased fixed (non-recurring) costs as a consequence of expanding requirements. The shift also yields important benefits in manufacturability (as distinct from design and development) and, potentially (this is an important caveat), for configuration management and update. It also can enable dramatic improvements in performance and responsiveness, since automation can often respond much more quickly than is possible in the scope of human reaction times.

The integrations -- and the ongoing shift in roles among hardware, software, and humans -- create significant challenges for modeling and analysis at every level of design consideration, from requirements and architecture to low-level component design and implementation. Diverse modeling techniques are required, and the analyses associated with each yield fragments of evidence that contribute to an overall

assurance judgment. Additionally, the software designs are generally more complex because they must increasingly embody resiliency with respect to faults (and adverse actions) from hardware sensors and affectors, from human operators, from human users, and from other systems that are interconnected to support systems-of-systems functionalities.

THE MODERN OPERATING ENVIRONMENT INVOLVES THE INTERLINKING OF MULTIPLE SYSTEMS

This interlinking and its benefits are identified with phrases such as *System of Systems* and *Net-Centric*, which suggests a kind of pervasive interoperation among systems. The benefits are amply described in the literature envisioning future systems that provide commanders, for example, with the benefits of geo-dispersed sensor networks, multiple analytic capabilities, and a selection of fires enabling choice according to immediate tactical needs rather than the position of a particular platform at the moment.

The interoperation can have homogeneous and heterogeneous features, enabling, for example, coalescing of data from dispersed sensors of a particular kind as well as the use of sensors, say, from a range of kinds of platforms. Indeed, this approach, of creating functionalities that involve multiple systems operating together, diminishes emphasis on the design of fully self-integrated platforms and increases emphasis on creation of distinct capabilities that build on more general-purpose basic platforms. This shift in emphasis has been highlighted, for example, by ADM Greenert in his presentations as a focus on payloads over platforms, where the "payload" is the assembled capability. This also suggests the possibility that payloads can be rapidly assembled and adapted according to changing mission needs.

The system-of-system approach can also involve use of mobile devices, forward-deployed capabilities for managing and analyzing large amounts of data, and the integration of diverse civil, coalition, and international partners.

SUPPLY CHAINS ARE COMPLEX, EXTENSIVE, AND GEOGRAPHICALLY DIVERSE

Relatively fewer software-reliant systems are constructed in an entirely greenfield fashion. These cases include primarily certain narrow categories of highly critical systems (such as high-grade crypto) and particular embedded controls (such as flight controls -- vs. the broader category of avionics).

Most other systems involve components from diverse sources. These other systems include web applications, data analytics, command and control systems, network management systems, data analysis systems, engineering design systems, and so on. Indeed, embedded controls, formerly frozen into a firmware snapshot, are now increasingly network enabled and supportive of update and reconfiguration. This includes, for example, the firmware on storage devices (such as disk drives and SSDs) as well as a wide range of embedded controls. This phenomenon has been labeled the Internet of Things (IoT), and, as well documented, it offers great potential benefits, but also some risks and uncertainties. In this regard, the technology of cyber-physical systems is evolving rapidly, and the market space is adapting accordingly. Purely physical sensors may now embody computational capability and embedded software components.

Vivid examples of the richness of supply chains for modern systems are web applications, mobile devices, and data-analytic applications. These all typically involve multiple software frameworks, libraries, code generators, and various kinds of supporting tooling in the development process (what Microsoft calls "managed code"). A small government web application recently evaluated by several of the RT-119 faculty leaders included more than a dozen different external components, and this is typical for web

applications. These external components can be vendor subsystems, open source components, or custom subcontracted components.

The ecosystems for modern mobile devices (IOS and Android, principally) each include more than a million different available applications from thousands of different sources world-wide.

From the perspective of assurance, these rich supply chains offer both challenges and, potentially, opportunities. The challenge is that components are often provisioned in a manner that precludes direct analysis. This is often by design, since suppliers seek to protect their intellectual property through techniques such as obfuscation and provisioning only executables. The opportunity is that (1) when there are compositional evaluation techniques and (2) components are used only when they have previously been evaluated (and fingerprinted), then any assurance certification can benefit multiple systems.

MODERN SYSTEMS ARE MORE AUTONOMOUS AND ACTIVELY RESILIENT

As noted above, roles within systems are shifting from hardware components and from human operators into software. For example, over the past three to five decades, aircraft and avionics designs have shifted from human operators managing every decision related to aircraft control surfaces to a state of potential full autonomy.

With an increase in autonomy, the demands on assurance practice necessary also increase, both due to the likely added complexity and also due to the fact that engineers become increasingly responsible to assure that rules of engagement are respected, whereas previously this was the responsibility of human operators.

SYSTEMS ARE UNDER CONTINUOUS ATTACK

The attacks may be both network-based attacks, for example during deployment and operation, and also supply-chain attacks, during development and sustainment.

With the growing sophistication of attackers, it is increasingly important for systems to be designed for resilience, and to degrade gracefully when partially compromised. Attackers can compromise sensors and system hardware as well as software, which complicates the definition and evaluation of resilience.

The increasing sophistication of attackers -- and the rapid evolution of the threat environment -- place greater responsibility on both system designers and assurance evaluators to address resilience and security issues effectively at all levels of design, from requirements and architecture to implementation and operation. Adherence with the engineering-related features of best-practice security controls (for example, http://www.cisecurity.org/critical-controls.cfm for mainstream civilian systems) must be addressed as part of the overall assurance practice.

Challenges of Practices

The second category of challenges related to the rapid evolution of the technology infrastructure. We identify five categories of challenges related to current practice, changes in the mission environment, and the evolving technology environment.

MODERN NATIONAL SECURITY SYSTEMS HAVE GREATER, GROWING COMPLEXITY

An increase in complexity is a natural consequence of the combination of [a] rising ambitions regarding system functionality and adaptiveness and [b] the ability of the technical community to deliver in response to those rising ambitions. This unboundedness, which is a key characteristic of software, has the consequence of growth in complexity. Addressing the complexity through composition in both models and systems structure is therefore a primary consideration. Rich supply chains, as noted above, are a key enabler to the growth in both capability and complexity -- and also flexibility.

A feature of this complexity, resulting from the complex architectures, the now-rich supply chain structure, and a consequent reliance on multiple diverse sources, is that there are *trust gradients* within a system. That is, there are attack surfaces at internal APIs and service interfaces.

The complexity also arises from particular features of modern development practice including: (1) Systems incorporating autonomy and resiliency, through which control loops and self-adaptation are integrated into systems, giving rise to architectures whose structure is dynamic rather than static, (2) High performance algorithms that rely on randomness and stochastic behavior, both as integral to algorithm design and as a consequence of multi-threading, (3) Concurrency and distribution, as the only means to scale performance, (4) Use of cloud and cloud-like resources for computation and data-management infrastructure, and, as noted above, (5) Reliance on rich supply chains for libraries, frameworks, generators, and specialized components.

For example, in the case of asynchronous concurrent code, such as would be used to exploit multi-core and GPU platforms, non-deterministic erroneous behavior could be perceived as acceptably rare in a test process, but commonly recurring in deployment. Suppose a data race manifests, say, every 1,000,000 runs of a test case. This means that the correct result of a single test will be given with exceedingly high likelihood (99.999%). But if the same code runs every millisecond on a small cluster of, say, a thousand cores, then on the average the error would manifest once each second.

MODERN NATIONAL SECURITY SYSTEMS HAVE REQUIREMENTS FOR HIGHER LEVELS OF ASSURANCE

There are two drivers of the need for higher levels of assurance.

First, systems are taking over an increasing span of mission roles, and so the consequences of failures and attacks can be much greater. Examples of drivers of this growth include reliance on autonomy, the need for resilience and adaptivity, and the benefits of interconnecting systems with other systems. Each of these amplifies the potential extent of consequence of a failure or a successful attack. And, indeed, the resilience that needs to be built in to enable continued operations in a compromised environment adds further to the complexity.

Second, adversaries are becoming increasingly sophisticated, with more subtle and patient modes of attack, including supply-chain attacks during development as well as network-delivered attacks during deployment and operation.

These points are related to the first challenge of complexity -- individually, each of the challenges significantly raises the bar for assurance. Taken together, they highlight the urgency of a strategic approach to assurance.

PROCESS COMPLIANCE DOES NOT ASSURE QUALITY

Process discipline, process-related key performance areas, and process metrics are essential to support effective management of the many steps required to develop, assure, sustain, and evolve systems. Indeed, many standards of engineering practice focus on adherence to particular kinds of process discipline. Correlations are demonstrated that show "mature" processes associate with higher levels of quality. Nonetheless, it is important to understand that process discipline does not in itself lead to high levels of quality or necessarily advance the *ilities* appropriate to the system being developed. No amount of process discipline can substitute for the direct evaluation of system-related artifacts: sensors, actuators, software code, models, test cases, analysis results, and so on.

Good process guidance for security, such as the SDL (Howard and Lipner) and the broader BSIMM framework (McGraw et al.), identify appropriate points in the process for constructing models and undertaking such direct assessment of artifacts. A careful examination of these process frameworks reveals that assurance judgments unavoidably come from the evidence created around the artifacts, not from the process discipline on its own or from metrics and other abstracted proxies.

Indeed, when process compliance is substituted for direct assessment, there can be inappropriate incentives, related to the legal concept of "safe harbors," created to actually avoid direct evaluation of artifacts and direct knowledge of their shortcomings. This is dangerous, since there is no substitute for the accumulation of necessary artifact-based evidence and the direct assay of that evidence.

MANY CURRENT PRODUCT-FOCUSED ACCEPTANCE EVALUATION PRACTICES ARE NOT SUFFICIENTLY EFFECTIVE, PARTICULARLY BLACK-BOX TESTING

Current product-focused techniques also present challenges. Application of many direct techniques at the system level or for major subsystems may be limited to "black box" experimentation with these large components. This can provide strong indications, but it cannot provide strong assurance except when certain restrictions are respected regarding the nature of the engineering designs implemented within the opaque components. More invasive "white box" or "gray box" techniques may be avoided in order to prevent exposure of developer intellectual property to customers and third-parties. When the components have even modest complexity, this can create risk of weak conclusions that do not support strong assurance judgments.

MANY COMMON PRACTICES ARE OUT OF PHASE WITH MISSION AND ASSURANCE REQUIREMENTS

Modern systems engineering involves integrations of diverse components and, often iterative and incremental processes for development. In commercial small-team development, test cases and test scaffolds are developed concurrently with the system being tested. And diverse other quality-evaluation techniques are employed including inspection, sound and heuristic static analysis, and various kinds of dynamic analysis, logging, and monitoring. Integration and evaluation is accomplished in steps, often with the customer directly involved in the process.

Adoption of these techniques can be problematic when there are arm's length relationships between suppliers and customers, including not just government and primes, but also primes and their vendors and subcontractors. The arm's length relationships can create barriers both to the necessary direct evaluation and also to incremental customer-involved risk-managed approaches to developing and

assaying assurance-related evidence. (See, for example, Boehm, et al., *The Incremental Commitment Spiral Model*. Addison-Wesley 2014.)

This is complicated by the fact the much of the tooling and technology in support of present assurance practices can provide only heuristic associational results rather than sharp conclusions. Additionally, the current practices tend to emphasize acceptance evaluation processes based primarily on whole-system after-the-fact testing and evaluation. These practices are not only costly and imprecise (i.e., often unable to come to definitive conclusions, particularly with respect to security criteria), they also pose great challenges when systems are evolving (in response to changing mission context and infrastructure) and recertification is required.

OVERALL ASSURANCE JUDGMENT INCORPORATES A WIDE AND EXPANDING RANGE OF DISTINCT QUALITY CRITERIA

These criteria can relate to the many dimensions of security (confidentiality, integrity, availability, non-repudiation, etc.), reliability (fault-tolerance, resiliency and response in compromised environments, etc.), safety (with respect to hazards to operators and others, etc.), responsiveness (behavior under load, etc.), usability (by human operators and others interacting with the system), interoperation (compatibility and support for with SoS APIs and practices), as well as a diverse range of *ilities* (evolvability/extensibility, maintainability, portability, etc.).

APPROACH

As noted above, the potential exists now, in emerging technology and tools, to obtain assurance-related conclusions more readily and that, with appropriate integration into the full span of development process, tooling, and modeling, could potentially support improved assurance including:

- More confident judgments
- Lower cost and uncertainty at the time of final acceptance evaluation (OT&E)
- Better understanding and weighting of the diverse and particular quality criteria, including an understanding and direct modeling of the contingencies that relate the various constituent assurance judgments
- Evidence-based judgments that preserve dependencies and rationale for critical assurance-related development decisions (prospects for DT&E in support of OT&E),
- And, by combining the elements above, prospects for lower-cost incremental recertification for evolving systems.

The purpose of the task is to explore barriers and opportunities to achieving this vision. The team members recognize that, on the basis of recent and emerging technical developments, it is now timely to develop and integrate a set of key ideas related to assurance. The activity is structured to support effective validation of the problem, to assess viability of a vision of evidence-based assurance from technical and process perspectives, and to advance a series of incremental steps towards that vision.

The first year of effort in the RT-119 task focused in four specific areas. It is a management principle in the task organization that every team member should be involved in all four threads of the effort:

Baseline Analysis of Codified Best Practices

The baselining analysis for existing practices has enabled the team to identify existing assurance practices as set forth in existing instructions and standards. The initial choice of existing material for review was made by the team based on experience and on informal input from sponsors. The baselining exercise provides a framework for assessing validity and significance of any proposed improvements in assurance practices. A key feature is on identifying meta-criteria (see below). More specifically, the focus of this part of the effort is in five areas:

> [a] Preliminary identification of a set of meta-criteria to be applied both in evaluation of identified existing practices, as a baseline, and also for any proposed interventions that arise from the research areas included in this task.
>
> [b] Identification of a representative set of practices, as a preliminary effort in anticipation of a joint selection. Evaluation of those practices according to the preliminary set of meta-criteria, with a focus on the salient aspects of the practices, rather than on detailed evaluation of the mechanics of the practice.
>
> [c] Revision of the meta-criteria according to the experience of evaluating the baseline practices
>
> [d] Evaluating additional practices, as guided by the joint selection by sponsor and team of particular practices for consideration.

[e] Continuing to iterate with respect to both meta-criteria and practices. Additionally, apply the meta-criteria to emerging techniques as developed in the other task areas.

The teams have already completed step (1) and parts of steps (2) and (3) above. The practices currently being considered include:

[a] DO-178B/C, DO-333. This is a relatively stringent set of technical criteria for evaluation of flight controls, which have extreme requirements for reliability and safety

[b] ISO 15408, Common Criteria. This is implemented in the US through the NIAP process. This is a practice focused on security evaluation for critical systems components. It is based on ideas in the 1983 "Orange Book" Trusted Computer System Evaluation Criteria (TCSEC), but with significant improvements enabling third-party evaluations and standard definitions of certain security functionalities ("protection profiles").

[c] IEC 62304, ISO 14971. Practices used for embedded medical device software lifecycle management. This practice focuses on safety-critical embedded software. This work is

[d] DoDI 8500.02, STIGs, and related standards. This practices focuses on cybersecurity risk management.

Identification and Validation of Meta-Criteria

An initial evaluation of these practices led to a preliminary set of meta-criteria according to which any assurance evaluation practice could be assessed. The evaluation above has resulted in improvements to the initial inventory of meta-criteria, but the process of identifying and refining the meta-criteria remains a work in progress.

The meta-criteria, which derive from the initial consideration identified standards-based best practices, are used to provide a preliminary early validation for particular technical methods for evaluation (see below) and also for potential new best practice concepts.

Advancement in Technical Practice

Advancement in technical practice is at the core of this task effort. This work relates directly to subtasks 4, 5, and 6 (requirements and validation, architecture and resilience, technical methods for modeling and analysis) and, in future effort by the team, subtasks 2, 3, and 7.

Concepts for Improved Best-Practice Standards

Building on the emergent meta-criteria and results related to advancing the technical practice, the task team is identifying concepts for improved best-practice standards. This aspect of the effort is planned primarily for later years of the task execution.

The sections below highlight some of the results of the first year of effort. A principal result is the synthesis of the meta-criteria. The meta-criteria resulted from the joint consideration of (1) the current baseline of evaluation best-practices, as codified into ratified evaluation standards, (2) consideration of emerging technical means for modeling, analyzing, and evaluating systems, including support for argumentation

and dependency management, (3) consideration of the mission challenges identified above, (4) consideration of the emerging systems-related challenges identified above, and (5) some initial consideration of prospects for new approaches to assurance that could provide greater confidence, lower cost, easier recertification, and other benefits.

Meta-Criteria

The initial inventory includes 17 meta-criteria:

- Specific technical **quality attributes** addressed and overall level of quality attainable and assurable for each.
- Reliability/**validity** of results – from sound verification to heuristic correlates.
- Phases of **process** where evaluation activities are undertaken – ranging from early (requirements and architecture) to after-the-fact.
- Access required by evaluators to supplier **intellectual property** and artifacts.
- Role of evaluation considerations in **architectural decisions** and implementation choices.
- Role of **process indicators** versus direct examination of development artifacts.
- Reusability of **evidence** from prior evaluations for incremental re-evaluation and recertification – status of evidence produced.
- Diversity of **kinds of evidence** to support judgments – kinds of models, informal/formal, linkage and traceability, etc.
- Up-front investment (**tooling**, **training**) and ongoing **cost** (based on complexity and scale).
- Benefits to cost and schedule; enhancements to **engineer productivity** and risk management.
- **Composability** of results for **components**, libraries, and frameworks in evaluating aggregates. **Scaling** potential and limitations.
- Support for **ecosystems**: mobile devices, big-data analytics, graphical interaction, etc.
- **Inter-rater reliability** in evaluation, including ability to assess and extent of existing assessment.
- **Incentives for developers** to produce evidence to be used by evaluators.
- **Incentives for evaluators** to publish evidence back to developers and to end customers.
- **Risks of incorrect assessments** and assurance judgments.
- **Skill requirements** and training needs for evaluators.

As noted above, the team evaluated several established practices, as codified in identified standards, for developing and supporting assurance judgments. For each, there is some detailed analysis with respect to the meta-criteria. This report includes brief summaries for two of the evaluations. These short summaries are intended primarily to suggest how the development of meta-criteria is influenced by this baselining process. In particular, the purpose is not to offer any judgments regarding the efficacy of these particular standards.

Because of the role of this context, these results are not intended for dissemination. Below are short summaries related to three of the several standards suites being evaluated in the baselining activity. (Summaries of the principal features of these standards are widely available.)

Common Criteria [CC]

CC (version 3.1, revision 4) describes certification requirements and process regarding security properties. It is an international standard (ISO 15408) administered by national authorities. In the US, the CC is administered by the National Information Assurance Partnership (NIAP), which is a joint activity of NSA and NIST that manages the CC Evaluation and Validation Scheme (CCEVS).

CC describes a general framework for evaluating software artifacts against security requirements, such as encryption, password authentication, and information flow integrity. It embraces critical security concepts such as reference monitor, trusted computing base (TCB), etc.

Much of the CC evaluation process focuses on textual documents including requirements and designs, with many checks of text documents for content and presentation. Formal methods may be used at the specification level for high assurance levels such as EAL 7, but it is important to note that implementation artifacts are not evaluated beyond sampled testing. As a consequence, even at the highest assurance level, there is no formal or sound proof that the implementation meets the specification. Releasing code to evaluators (for using automated tools) is a stated goal of CC at high assurance levels, but it does not appear to be enforced. Code-level analysis techniques and tools do not appear to be covered in the standard. It is unclear how inter-evaluator reliability is assessed.

The evaluation in made on a snapshot of the system and its design documents, and the evaluation is generally performed after the fact. Certificates appear to remain valid when products are revised. In some cases severe security vulnerabilities are discovered later. This was noted in the Heartbleed episode: "If your product uses an affected version of OpenSSL and it has already been certified or validated, you are not required to recertify. However, we would strongly urge you to demonstrate to your customers, potential customers and other stakeholders that you are committed to information security, and have thoroughly reviewed if your product or system is vulnerable to the Heartbleed bug. You should be prepared to take whatever steps are necessary to remediate the situation, up to and including recertification."

Re-certifications require significant effort. Although evidence developed by the third-party evaluators from earlier evaluations may be reused, the incentives to do this are mixed, since evaluators are not required to release evidence developed during an evaluation process.

CC does not have a mechanism to leverage the use of development documentation, formal methods, and tools to directly support evaluations. In particular, more formal methods cannot replace the burden of producing textual evidence. (For example, CompCert would not pass EAL1 certification without additional text-based documentation.)

At a conceptual level, CC supports composed judgments, at least for coarse-grained components (OS and DB) and require a per-composition snapshot evaluation. On the other hand, there is no explicit support for compositional reasoning of the kind needed for ecosystems such as for mobile, big-data, browser plug-ins, etc.

DO 178C

DO 178C and related standards (DO 178B, DO-333) are intended to support software evaluation for airborne systems by agencies such as FAA and other national authorities. DO178C describes a certification process for all commercial software-based aerospace systems. It is approved by FAA, EASA, Transport Canada, etc. It is recognized as an “acceptable means, but not the only means, for showing compliance with the applicable airworthiness regulations for the software aspects of airborne systems and equipment certification," with a purpose to fulfill a need for "industry-accepted guidance for satisfying airworthiness requirements" regarding both functionality and safety. Specific safety requirements are delegated to related standards (SAE ARP 4754 and 4761).

The DO178C standard emphasizes documentation of traceability among process stages, leading to "100% coverage" with testing. The premise is that software failures related to safety often arise from incorrect requirements, missing requirements, and unintended behavior. This motivates the emphasis on safety analysis, management of traceability/dependency, and test coverage.

The standard stratifies safety levels according to failure conditions, ranging from catastrophic to "no safety effect."

The standard levies requirements to produce documentation of software aspects of certification along with plans for software development, verification, configuration management, and quality assurance.

DoD Instructions and Directives

We undertook a preliminary analysis of DoD instructions and directives on acquisition to identify software engineering assurance requirements. The policies studied include DoDD 5000.02, DoDI 3000.09, DoDI 8500.01, NIST SP800-39, CNSSP No. 22, and the Application Security STIGs. The key results are that the general policies include links to specific guidance, and a lower-level policy, the Application Security STIG, includes many of the fundamentals needed to holistically build assurance cases. The security properties are documented through plan documents maintained by the program manager, such as the System Security Plan, Application Configuration Guide, and Standard Environment artifacts; the reliability and validity, and phases where evaluation occur, is driven by threat modeling; there are multiple artifact-based assurances that use tools and testing practices to reach conclusions; and third-party code and dependencies must be evaluated.

Architecture

The faculty leads for the architecture subtask are David Garlan and Bradley Schmerl. The architecture effort has three areas of focus:

(1) Improving resiliency, including tools to improve the effectiveness and analyzability of run-time adaptation.

> This includes repair generation using Stochastic Multiplayer Games (SMGs) to assure maximum system utility; latency aware, proactive approaches that assure timely adaptation and maximizes accrued system utility; and reasoning about mixed human and autonomous adaptation using SMGs. These three approaches add to the diversity of evidence for reasoning about resiliency for dynamic systems.

(2) Improving quality.

> [a] Utility-based reasoning techniques in multi-dimensional quality attribute space, e.g., balancing security, cost, and performance concerns
>
> [b] Design space exploration using model generation (using Alloy) explicitly addressing quality concerns, e.g., time-fidelity trade-offs software composition
>
> [c] Domain specific design environments for end-user architects that allow quality analysis, e.g., "e-Scientists" composing tasks and reasoning about performance
>
> [d] Formal analysis of frameworks to understand QA guarantees, e.g., security in Android.

Each of these four is based on explicit statement of technical quality attributes.

(3) Improving software and systems engineering.

> [a] Multi-model design techniques and tools which relate software, physical, network, electrical models to each other and check consistency and analysis dependencies
>
> [b] Modularization techniques for hybrid systems specifications, which improve reuse and incremental development of models.

These approaches are intended to promote composability of results.

This report highlights four principal results related to architecture:

Improving the link between software architecture and systems engineering. Modern cyber-physical systems interact closely with continuous physical processes like kinematic movement. Software component frameworks do not provide an explicit way to represent or reason about these processes. Meanwhile, hybrid program models have been successful in proving critical properties of discrete-continuous systems. These programs deal with diverse aspects of a cyber-physical system such as controller decisions, component communication protocols, and mechanical dynamics, requiring several programs to address the variation. However, currently these aspects are often intertwined in mostly monolithic hybrid programs, which are difficult to understand, change, and organize. We laid the

foundations for using architectural models to provide component-based benefits to developing hybrid programs. We have built formal architectural abstractions of hybrid programs and formulas, enabling analysis of hybrid programs at the component level, reusing parts of hybrid programs, and automatic transformation from views into hybrid programs and formulas. We evaluated our approach in the context of a robotic collision avoidance case study.

This work is reported in the paper:

> Ivan Ruchkin, Bradley Schmerl and David Garlan. Architectural Abstractions for Hybrid Programs. In Proceedings of the 18th International ACM Sigsoft Symposium on Component-Based Software Engineering (CBSE 2015), Montréal, QC, Canada, 4-8 May 2015.

Modeling for cyber-physical systems architecture. Cyber-physical systems (CPS) are heterogeneous, because they tightly couple computation, communication and control along with physical dynamics, which are traditionally considered separately. Without a comprehensive modeling formalism, model-based development of CPS involves using a multitude of models in a variety of formalisms that capture various aspects of the system design, such as software design, networking design, physical models, and protocol design. Without a rigorous unifying framework, system integration and integration of the analysis results for various models remains ad hoc. In this paper, we propose a multi-view architecture framework that treats models as views of the underlying system structure and uses structural and semantic mappings to ensure consistency and enable system-level verification from that of the models in a hierarchical and compositional manner. This was reported in:

> Akshay Rajhans, Ajinkya Y. Bhave, Ivan Ruchkin, Bruce Krogh, David Garlan, Andre Platzer and Bradley Schmerl. Supporting Heterogeneity in Cyber-Physical Systems Architectures. In IEEE Transactions on Automatic Control, Vol. 59(12):3178--3193, December 2014.

Modeling human interactions with systems. Stochastic multiplayer games can provide assurances for adaptation and planning of software systems that can cooperate with humans. Self-adaptive systems overcome many of the limitations of human supervision in complex software-intensive systems by endowing them with the ability to automatically adapt their structure and behavior in the presence of runtime changes. However, adaptation in some classes of systems (e.g., safety-critical) can benefit by receiving information from humans (e.g., acting as sophisticated sensors, decision-makers), or by involving them as system-level effectors to execute adaptations (e.g., when automation is not possible, or as a fallback mechanism). However, human participants are influenced by factors external to the system (e.g., training level, fatigue) that affect the likelihood of success when they perform a task, its duration, or even if they are willing to perform it in the first place. Without careful consideration of these factors, it is unclear how to decide when to involve humans in adaptation, and in which way. We are investigating how the explicit modeling of human participants can provide a better insight into the trade-offs of involving humans in adaptation. We contribute a formal framework to reason about human involvement in self-adaptation, focusing on the role of human participants as actors (i.e., effectors) during the execution stage of adaptation. The approach consists of:

(i) a language to express adaptation models that capture factors affecting human behavior and its interactions with the system, and

(ii) a formalization of these adaptation models as stochastic multiplayer games (SMGs) that can be used to analyze human-system-environment interactions. We illustrate our approach in an adaptive industrial middleware used to monitor and manage sensor networks in renewable energy production plants.

This work is reported in the following publications:

> Javier Cámara, Gabriel A. Moreno, David Garlan and Bradley Schmerl. Analyzing Latency-aware Self-adaptation using Stochastic Games and Simulations. 2015. Submitted for publication.
>
> Javier Cámara, Gabriel A. Moreno and David Garlan. Reasoning about Human Participation in Self-Adaptive Systems. In Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managed Systems (SEAMS 2015), Florence, Italy, 18-19 May 2015. To Appear.

Using stochastic search to support self-adaptation. This is an investigation of the application of stochastic search techniques to problems in the self-adaptive systems and architectural space. Specifically, we describe a genetic algorithm-based search, with a fitness function that leverages the probabilistic model checker PRISM, used in the planning phase of the MAPE loop of self-* systems. We demonstrate that such a planner can optimize for a variety of (sometimes competing, sometimes cooperating) quality attributes; can transition a system that is optimizing for one attribute to another; and can (hopefully) teach human operators unexpected or unknown things about how quality attributes interact and how to tune their self-* systems for various quality attributes. The attached short paper below uses initial results along these lines to launch a discussion about other potentially fruitful research directions along these lines.

> Zack Coker, David Garlan, Claire Le Goues , SASS: Self-adaptation using stochastic search. In Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managed Systems (SEAMS 2015), Florence, Italy, 18-19 May 2015. To Appear.

Requirements

DoD information assurance certification requires assigning appropriate security controls to graded information sensitivity levels: the more sensitive the information, or higher risk of exposure, the increase in mitigations needed. The ability to localize and trace information usage changes can limit the scope of recertification. We're extending a requirements specification language to express architectural, cross-component data flows with expressions for detecting conflicts due to cross-component flow restrictions and to enable flow tracing. In addition, we performed runtime analysis of the language and framework to measure performance on policies of increasing size and complexity.

> Daniel Smullen, Travis Breaux. "*Towards Rapid Re-Certification Using Formal Analysis*", 12th Annual Acquisition Research Symposium, May 2015.

We extended our work on "application profiles," which are mathematical descriptions of security requirements aimed at checking whether an application's requirements are consistent with a security policy.

The extension includes the ability to express data collection and transfer over an ontology of information types and needs, as well as, to trace data flow across profiles of multiple systems that share data to reason

about the compositions of services and the propagation of security requirements along data flows. The profiles are expressed in a specification language that has a formal semantics in Description Logic, which is suitable for automated inference to detect conflicts between requirements and policies.

This case study includes an evaluation based on a mobile app called Waze that employs third-party identification, storage and advertising services. The evaluation also includes a runtime simulation to assess the scalability of the approach using different theorem provers.

> Travis D. Breaux, Daniel Smullen, Hanan Hibshi. "*Detecting Repurposing and Over-collection in Multi-Party Privacy Requirements Specifications.*" To Appear: IEEE 23rd International Requirements Engineering Conference (RE'15), Ottawa, Canada, Sep. 2015.

Modular Analysis

Emerging results are in five areas:

[a] Sandboxing/encapsulation techniques and tools, including static assurance; dynamic monitoring; encapsulation

[b] Architecture-level assurance for quality attributes

[c] API protocol usability

[d] Concurrency libraries and security

[e] Directive mechanisms in mobile frameworks.

Three publications:

> *Preprocessor-Based Variability in Open-Source and Industrial Software Systems: An Empirical Study*. Accepted for Empirical Software Engineering (ESE), 2015.
>
> *Extracting Configuration Knowledge from Build Files with Symbolic Analysis*. Accepted at ICSE workshop, 2015
>
> *Searching the State Space: A Qualitative Study of API Protocol Usability*. Submitted

Looking ahead, the RT-119 task team are focusing modeling and analysis work in three areas:

(1) Augmenting process compliance with direct evidence. In summary terms, we can say that quality has three sources: [a] Process, which enables quality, [b] Engineering, which delivers quality, and [c] Evidence, which affirms quality. The RT-119 approach is to focus primarily on sources [b] and [c], and make recommendations regarding the best ways process (source [a]) can enable the necessary engineering and evidence production.

(2) Challenges and impediments to evidence-based approaches. An area of emphasis is the consideration of the interplay of technical mechanisms and business incentives. This interplay has several dimensions: [a] IP exposure and acceptance evaluation, [b] Safe harbors and incentives that can derive from a pure focus on process and compliance, and [c] False trade-offs among performance, cost (lifecycle, devt), quality, security, etc.

(3) Drivers of evidence-based approaches. The recent shifts in the mission and technology environments offer three principle enablers for evidence-based approaches: [a] Acquisition and sustainment, with growing emphasis on incrementality and evolution (in this regard, the recent INCOSE keynote speaker stated that "Systems engineering needs to shift focus from *built to last* to *built to evolve*"), [b] Structural realities of systems including dynamism (at the architecture level including resiliency and autonomy) and complex structure (frameworks, granular components, rich supply chains, as well as ecosystems for mobile, data, and vehicles), [c] Data-intensive modern tooling, including evidence structures and trade-off management supported by modeling, analysis, and dependency mappings among these. This includes consideration of diverse quality and security attributes, ilities, etc.

A key area of emphasis is to continue refinement of the meta-criteria and initiate validation with subject-matter experts (SMEs). This will include also interactions with SMEs experienced with the identified baseline assurance standards. In particular, the team will have completed evaluation of an initial set of baseline assurance practices, including refinement of the set of meta-criteria for evaluating both practice guidance/standards and also emerging technical practices (in the three areas of emphasis: architecture, requirements, implementation). These will lead to a preliminary concept document describing potential new-generation assurance practices and incentives.

The team will also continue to advance in the technical areas of focus including:

(1) architecture modeling and analysis, with emphasis on adaptivity, resiliency, and dynamism
(2) requirements specifications, analyses, and traceability
(3) composable implementation quality attributes, with focus on defect finding and repair, correct use of APIs and frameworks, and highly versioned systems.

This effort is significant because, in addition to adding cost and delay, many present assurance practices inhibit efficient modernization and evolution. The RT-119 Systemic Assurance project is developing data-intensive modeling and analysis techniques designed to directly integrate into development and evolution. It has a strong evaluation component, focused on identifying measurement capabilities and incentives for designed-in assurance, composability, and evolvability.